ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки

каф. ЕОМ

Система розпізнавання голоса в розумному будинку

СТ. ГРУПИ КІУКІ-21-2
МАКСИМЕНКО М.А.

КЕРІВНИК
СТ.ВИКЛАДАЧ РАДЧЕНКО В.О.

Мета та задачі

Мета кваліфікаційної роботи розробити систему розпізнавання голоса в розумному будинку.

Задачі:

- Запропонувати архітектуру систему домашньої автоматизації з вбудованим розпізнаванням мовлення
- Побудувати модель перетворення мовлення на текст
- Розглянути принципи підключення пристроїв
- Розробити принципову схема розумної розетки
- Розробити програмний прототип керування розумним будинком за допомогою голосових команд

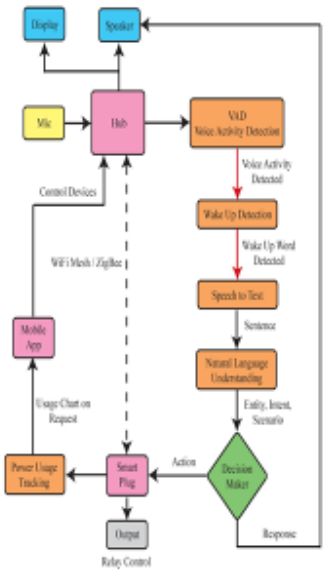
Архітектура системи

Запропонована система – це недорога, автономна та універсальна система домашньої автоматизації з вбудованим розпізнаванням мовлення та виявленням намірів. Її метою є спрощення системи та зниження її вартості. Це усуне потребу у високопродуктивних хмарних обчисленнях, що, у свою чергу, зменшить ризики для конфіденційності та кібербезпеки.

Система також є гнучкою, що дозволяє легко додавати до мережі нові прилади інших виробників для безпечної та надійної роботи.

Крім того, система включає розумні розетки з покращеним підключенням на відстані та можливостями вимірювання енергії.

На рисунку показано архітектуру системи, що складається з розумного концентратора та розумної розетки.

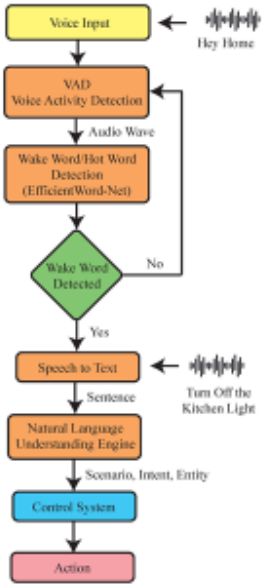


Розумний HUB

У системі домашньої автоматизації розумний хаб служить центральною точкою керування для взаємодії між компонентами. Може бути один або декілька хабів розумного дому, або їх може бути взагалі немає. Зазвичай розумні хаби покладаються на хмарну обробку для обробки потреб голосових помічників, що може призвести до непрацездатності, якщо хмарні сервіси недоступні. Щоб подолати це, розумний хаб Home оснащений вбудованим голосовим інтерфейсом користувача, який дозволяє користувачам спілкуватися із системою без потреби у сторонніх сервісах.

На рисунку показано огляд системи розпізнавання мовлення, що використовується в . Система складається з чотирьох компонентів: моделі виявлення голосової активності (VAD), моделі виявлення слів, що прокидаються, моделі перетворення мовлення на текст та моделі розуміння природної мови.

Модель VAD має алгоритм, який постійно контролює навколишнє середовище на наявність мовних сигналів та використовує VAD для ідентифікації сегментів мовлення з меншим споживанням енергії. Після виявлення озвученого сегмента спрацьовує модель виявлення слів, що прокидаються. Якщо слово, що прокидається, модель перетворення мовлення на текст перетворює аудіосигнали на речення. Механізм розуміння природної мови визначає сценарій, намір та сутність з речення. Це передається системі керування для прийняття рішення.

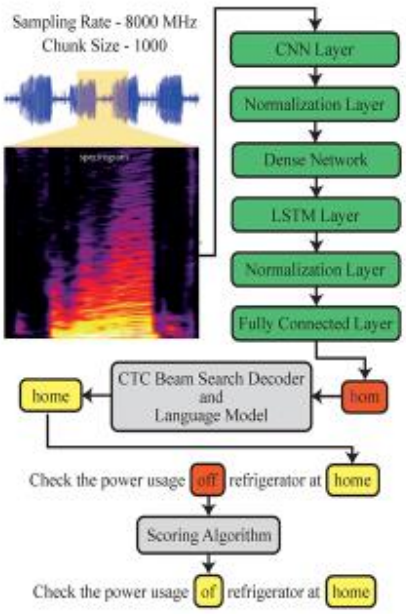


Модель перетворення мовлення на текст

Метою моделі перетворення мовлення в текст (STT) є аналіз аудіосигналу, його розбиття на частини, оцифрування у машинозчитуваний формат та створення найбільш відповідного текстового представлення. Для досягнення цієї мети системи перетворення мовлення в текст спираються на два типи моделей: акустичні моделі та мовні моделі.

Наша реалізація STT використовує просту згорткову нейронну мережу (CNN), як показано на рисунку.

Модель передбачає літери, вимовлені користувачем, і як тільки між словами настає пауза, передбачені літери пропускаються через декодер променевого пошуку конекціоністської часової класифікації (CTC) з обмеженнями лексики та мовною моделлю для отримання найкращої оцінки речення. Модель була навчена на наборі даних vosk-model-uk-v3



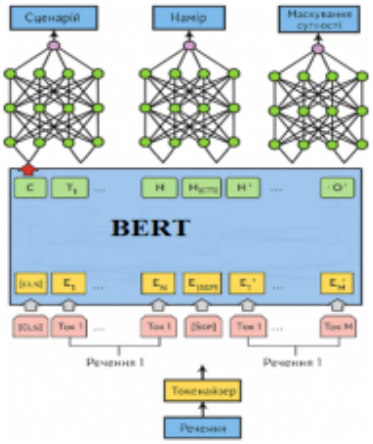
24.06.2025

Модель розуміння природної мови.

Наша система використовує попередньо навчену модель двонаправленого кодування з трансформаторів (BERT). Огляд цієї моделі показано на рисунку.

Вона витягує маску з вхідного речення, яка потім надсилається до штучної нейронної мережі (ШНМ), яка визначає сценарій, намір та сутність з вхідного речення . Прикладом цього може бути речення «Вимкніть світло на кухні», де сценарій – «IoT», намір – «ВИМКНІТЬ», а сутності – «Кухня: місцезнаходження» та «світла: пристрій».

Ці змінні потім можна використовувати в логіці керування для виконання потрібного завдання. Модель була навчена з використанням даних з відкритого коду.



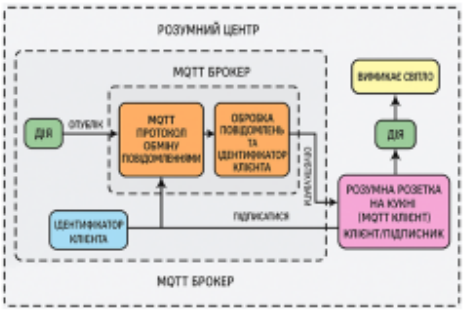
24.06.2025

Підключення

MQTT – це стандартний протокол обміну повідомленнями для Інтернету речей, розроблений Організацією з розвитку стандартів структурованої інформації (OASIS). Огляд протоколу показано на рисунку.

Це широко використовуваний протокол обміну повідомленнями в галузі Інтернету речей завдяки своїй простоті та функціям безпеки. Він має низькі накладні витрати як на кодування, так і на мережевий трафік.

Він використовує шифрування транспортного рівня (TLS) та автентифікує клієнтів за допомогою протоколу авторизації відкритого стандарту (OAuth) для забезпечення безпечного зв'язку між пристроями



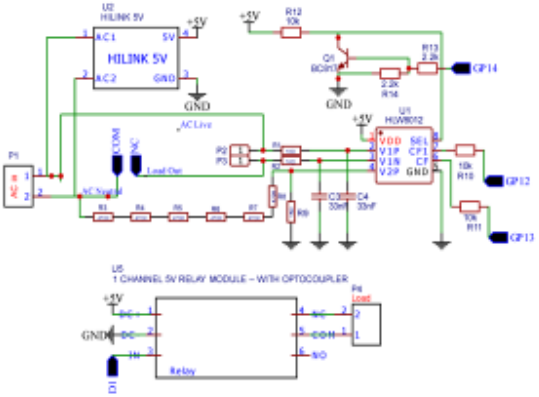
Розумна розетка

Для розумної розетки було обрано мікроконтролер ESP32 NodeMCU для збору даних датчиків та його роботи як контролера. Необхідний код для роботи реле та вимірювання енергії було написано за допомогою інтегрованого середовища розробки (IDE) Arduino.

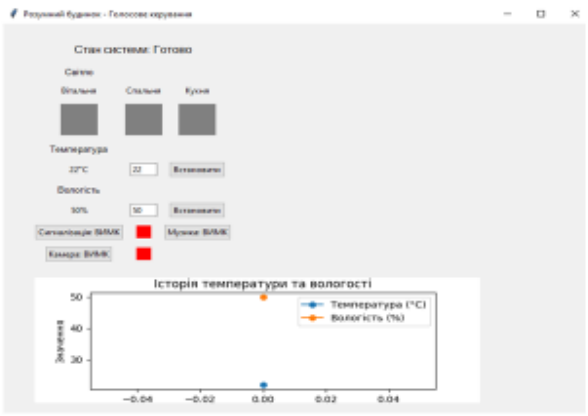
Розумна розетка живиться від модуля живлення постійного струму 5 В. Для забезпечення додаткової безпеки в схему керування реле включено запобіжник.

Розумна розетка використовує плату HLW8012 для вимірювання енергії. Для розумної розетки було розроблено схему та друковану плату, де всі модулі можуть поміститися в невеликому просторі.

Принципова схема розумної розетки показана на рисунку. Розумна розетка призначена для моніторингу споживання енергії, коли вона не підключена до концентратора, та для передачі записаних даних, коли вона підключена до концентратора.



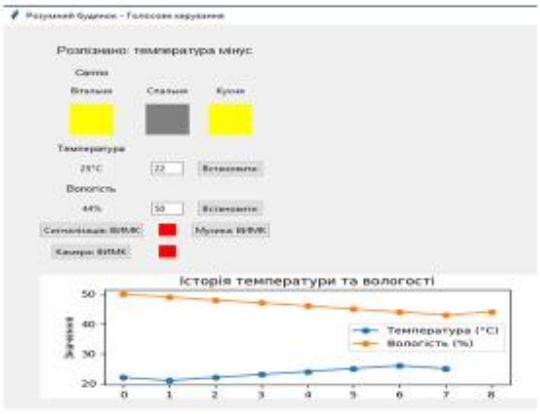
Головний інтерфейс для керування розумним будинком

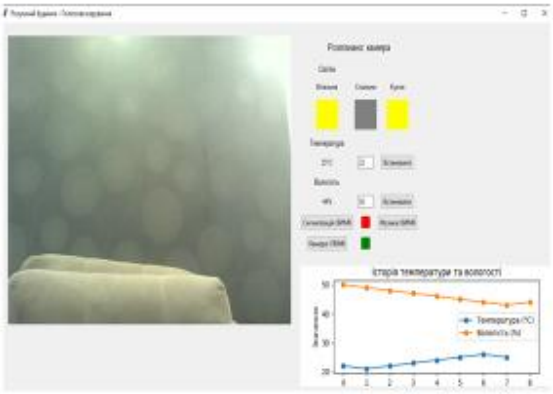


Інтерфейс панелі інструментів показано на рисунках показано система керування розумним будинком. Ця панель інструментів служить інструментом для керування пристроями та їх моніторингу і працює локально на розумному концентраторі

24.06.2025

Керування показниками вологості, температури та камерою





24.06.2025

Висновки

В ході роботи було розроблено систему розпізнавання голоса в розумному будинку.

Були вирішені наступні задачі:

- Запропоновано архітектуру систему домашньої автоматизації з вбудованим розпізнаванням мовлення
- Побудовано модель перетворення мовлення на текст
- Розглянуто принципи підключення пристроїв
- Розроблено принципову схема розумної розетки
- Розроблено програмний прототип керування розумним будинком за допомогою голосових команд

ДОДАТОК Б

ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

```
import tkinter as tk
from tkinter import ttk, messagebox
import speech_recognition as sr
from vosk import Model, KaldiRecognizer
import json
import pyaudio
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import threading
import pygame  # Додано для відтворення музики

# Ініціалізація моделі Vosk для української мови
model = Model(r"models\vosk-model-uk-v3")
recognizer = KaldiRecognizer(model, 16000)

# Ініціалізація pygame для відтворення музики
pygame.mixer.init()

class SmartHomeApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Розумний будинок - Голосове керування")

        # Стан приладів
        self.lights = {"living_room": False, "bedroom": False,
"kitchen": False}
        self.temperature = 22
        self.humidity = 50
        self.security = False
        self.music = False
        self.temperature_history = [self.temperature]  # Історія
температури для графіка
        self.humidity_history = [self.humidity]  # Історія
вологості для графіка

        # Головне вікно
        self.main_frame = ttk.Frame(root)
        self.main_frame.pack(pady=10, padx=10, fill="both",
expand=True)

        # Елементи головного вікна
        self.status_label = ttk.Label(self.main_frame,
text="Стан системи: Готово", font=('Arial', 12))
        self.status_label.grid(row=0, column=0, columnspan=3,
pady=10)
```

```
        # Керування світлом
        ttk.Label(self.main_frame, text="Світло", font=('Arial',
10)).grid(row=1, column=0, pady=5)

        # Індикатори світла у вигляді кіл
        self.light_indicators = {
            "living_room": tk.Canvas(self.main_frame, width=50,
height=50, bg="gray"),
            "bedroom": tk.Canvas(self.main_frame, width=50,
height=50, bg="gray"),
            "kitchen": tk.Canvas(self.main_frame, width=50,
height=50, bg="gray")
        }
        # Назви кімнат над індикаторами
        ttk.Label(self.main_frame, text="Вітальня").grid(row=2,
column=0, pady=5)
        ttk.Label(self.main_frame, text="Спальня").grid(row=2,
column=1, pady=5)
        ttk.Label(self.main_frame, text="Кухня").grid(row=2,
column=2, pady=5)

        self.light_indicators["living_room"].grid(row=3,
column=0, pady=5)
        self.light_indicators["bedroom"].grid(row=3, column=1,
pady=5)
        self.light_indicators["kitchen"].grid(row=3, column=2,
pady=5)

        # Керування температурою
        ttk.Label(self.main_frame, text="Температура",
font=('Arial', 10)).grid(row=4, column=0, pady=5)
        self.temp_label = ttk.Label(self.main_frame,
text=f"{self.temperature}°C")
        self.temp_label.grid(row=5, column=0, pady=5)
        self.temp_entry = ttk.Entry(self.main_frame, width=5)
        self.temp_entry.grid(row=5, column=1, pady=5)
        self.temp_entry.insert(0, str(self.temperature))
        self.set_temp_button = ttk.Button(self.main_frame,
text="Встановити", command=self.set_temperature)
        self.set_temp_button.grid(row=5, column=2, pady=5)

        # Керування вологістю
        ttk.Label(self.main_frame, text="Вологість",
font=('Arial', 10)).grid(row=6, column=0, pady=5)
        self.humidity_label = ttk.Label(self.main_frame,
text=f"{self.humidity}%")
        self.humidity_label.grid(row=7, column=0, pady=5)
        self.humidity_entry = ttk.Entry(self.main_frame,
width=5)
        self.humidity_entry.grid(row=7, column=1, pady=5)
        self.humidity_entry.insert(0, str(self.humidity))
        self.set_humidity_button = ttk.Button(self.main_frame,
text="Встановити", command=self.set_humidity)
```

```
        self.set_humidity_button.grid(row=7, column=2, pady=5)

        # Керування сигналізацією
        self.security_button = ttk.Button(self.main_frame,
text="Сигналізація: ВИМК", command=self.toggle_security)
        self.security_button.grid(row=8, column=0, pady=5)
        # Індикатор сигналізації
        self.security_indicator = tk.Canvas(self.main_frame,
width=20, height=20, bg="red")
        self.security_indicator.grid(row=8, column=1, pady=5)

        # Керування музикою
        self.music_button = ttk.Button(self.main_frame,
text="Музика: ВИМК", command=self.toggle_music)
        self.music_button.grid(row=8, column=2, pady=5)

        # Графік температури та вологості
        self.fig, self.ax = plt.subplots()
        self.canvas = FigureCanvasTkAgg(self.fig,
master=self.main_frame)
        self.canvas.get_tk_widget().grid(row=9, column=0,
columnspan=3, pady=10)
        self.update_graph()

        # Ініціалізація голосового двигуна
        self.mic = sr.Microphone()
        self.r = sr.Recognizer()
        self.listening_thread =
threading.Thread(target=self.start_listening, daemon=True)
        self.listening_thread.start()

    def update_gui(self):
        for room, indicator in self.light_indicators.items():
            color = "yellow" if self.lights[room] else "gray"
            indicator.config(bg=color)
        self.temp_label.config(text=f"{self.temperature}°C")
        self.humidity_label.config(text=f"{self.humidity}%")
        self.security_button.config(text=f"Сигналізація:
{'УВІМК' if self.security else 'ВИМК'}")
        self.music_button.config(text=f"Музика: {'УВІМК' if
self.music else 'ВИМК'}")
        # Оновлення індикатора сигналізації
        self.security_indicator.config(bg="green" if
self.security else "red")
        self.update_graph()

    def toggle_lights(self, room):
        self.lights[room] = not self.lights[room]
        self.update_gui()

    def toggle_security(self):
        self.security = not self.security
        self.update_gui()
```

```
    def toggle_music(self):
        self.music = not self.music
        if self.music:
            pygame.mixer.music.load("gutsulka-ksenya.mp3")  #
Завантажуємо трек
            pygame.mixer.music.play()  # Відтворюємо музику
        else:
            pygame.mixer.music.stop()  # Зупиняємо музику
        self.update_gui()

    def set_temperature(self):
        try:
            new_temp = int(self.temp_entry.get())
            if 10 <= new_temp <= 30:
                self.temperature = new_temp

self.temperature_history.append(self.temperature)
                self.update_gui()
            else:
                messagebox.showerror("Помилка", "Температура
повинна бути між 10°C та 30°C")
        except ValueError:
            messagebox.showerror("Помилка", "Введіть коректне
число")

    def set_humidity(self):
        try:
            new_humidity = int(self.humidity_entry.get())
            if 0 <= new_humidity <= 100:
                self.humidity = new_humidity
                self.humidity_history.append(self.humidity)
                self.update_gui()
            else:
                messagebox.showerror("Помилка", "Вологість
повинна бути між 0% та 100%")
        except ValueError:
            messagebox.showerror("Помилка", "Введіть коректне
число")

    def update_graph(self):
        self.ax.clear()
        self.ax.plot(self.temperature_history, marker='o',
label="Температура (°C)")
        self.ax.plot(self.humidity_history, marker='o',
label="Вологість (%)")
        self.ax.set_title("Історія температури та вологості")
        self.ax.set_xlabel("Час")
        self.ax.set_ylabel("Значення")
        self.ax.legend()
        self.canvas.draw()

    def process_command(self, text):
```

```
        text = text.lower()
        if "світло" in text:
            if "вітальня" in text:
                self.toggle_lights("living_room")
            elif "спальня" in text:
                self.toggle_lights("bedroom")
            elif "кухня" in text:
                self.toggle_lights("kitchen")
        elif "температура" in text:
            if "плюс" in text:
                self.temperature += 1

self.temperature_history.append(self.temperature)
            elif "мінус" in text:
                self.temperature -= 1

self.temperature_history.append(self.temperature)
            self.update_gui()
        elif "вологість" in text:
            if "плюс" in text:
                self.humidity += 1
                self.humidity_history.append(self.humidity)
            elif "мінус" in text:
                self.humidity -= 1
                self.humidity_history.append(self.humidity)
            self.update_gui()
        elif "сигналізація" in text:
            self.toggle_security()
        elif "музика" in text:
            self.toggle_music()
        self.update_gui()

    def start_listening(self):
        audio_stream = pyaudio.PyAudio().open(
            rate=16000, channels=1, format=pyaudio.paInt16,
input=True, frames_per_buffer=8192
        )
        while True:
            data = audio_stream.read(4096)
            if recognizer.AcceptWaveform(data):
                result = json.loads(recognizer.Result())
                command = result.get("text", "")
                if command:
                    self.status_label.config(text=f"Розпізнано:
{command}")
                    self.process_command(command)
            self.root.update()

if __name__ == "__main__":
    root = tk.Tk()
    app = SmartHomeApp(root)
    root.mainloop()
```